L3: Entry 8 of 70 File: USPT Apr 24, 2001

DOCUMENT-IDENTIFIER: US 6223343 B1
TITLE: Computer system and method to track and control element changes throughout application development

Application Filing Date (1):
19980403

Brief Summary Text (8):
FIG. 2 is an illustration of the basic components of main frame 2. In FIG. 2, main frame 2 includes three basic components: support platform 26, production platform 28, and developer platform 30. Support platform 26 is used, for example, to load new programs, operating systems and the like, to evaluate its use for the entire computer system. Developer platform 30 includes development tools for creating and testing new development programs. Production platform 28 then distributes the developed and tested programs/elements for distribution to the necessary components of the system.

Brief Summary Text (16):
PRJINFO: a facility that allows customers to perform various change management functions on groups of elements within an Issuance Name. Among these functions are the ability to build and mark ready ICMs, cross-reference elements both in development and production (PRODXREF), move elements to the system testing environment (ANTSISS), sign-out elements to an issuance name, and associate elements to a user-id.

Brief Summary Text (23):
A. Build and edit allows the entering of information which is needed to initiate an issuance. It contains a build and edict mode and will allow for a WEEKLY (cutoff Monday for implementation the following Monday), DAILY (requires 2 days lead time for implementation), EMERGENCY (to be implemented A.S.A.P.), DUMMY (secures components and produces analyst and shopwide documentation) and DUPLICATE (for advance pilot testing). The facility supports four types of issuances. They are as follows:

Brief Summary Text (32):
The issuance processor is a background job that will perform the following tasks to transfer a program (e.g., a ADF program, and the like) and all of its components from a test to production environment. The issuance processor is a batch job submitted from BLDISS when a component is marked "ready for issuance". The purpose of this job is to generate the proper load code and store the load and source into a secured production library. Documentation and audit trails are also produced at this time.

Brief Summary Text (60):
Analysts then retrieve the elements, alter or change the element, and test the changed element in step S4. Once the analyst is satisfied with the changed element, the limited number of modified elements are stored on an area panvalet in step S6. Issuance information about the changed element is provided to a librarian on a sheet of paper in step S8. The librarian creates the ICM indicative of the changed element using BLDISS in step S10, and further all the issuance information associated with the changed element. The basic information entered by the librarian includes what element needs to be issued, who is issuing the element, when does the element need

to be issued, and where does the changed element need to go for update purposes and why.

Brief Summary Text (61):
The analyst generally has the responsibility to do the actual software development, and the librarian is a clerical-type person that enters the specific element change information in the appropriate system, e.g., BLDISS. Thus, the analyst is the person who alters the element, and the librarian is the person who stores the data entry. The librarian uses BLDISS in step S10 to enter all the who, what, where and why information about the changed element. Once testing is completed, the element is stored on the area panvalet and the librarian creates the ICM, the librarian marks the ICM ready using BLDISS in step S12. The element source code will then be put into production via invocation of POSTISS in step S16. POSTISS creates executables for the source code, and distributes the executables as required in step S18.

Brief Summary Text (103):
Regional Support production is a valid destination for document issuances. Regional Support Test issuances is not a valid destination. Therefore, any document to be used for Regional Support should be built through BLDDOC, printed to hardcopy and sent to Regional Support. Regional Support will also be able to print off a hardcopy of the document through BLDDOC. An Issuance Control Member may be built for the document through BLDISS, but do not generally mark it "READY FOR ISSUANCE". By not issuing the document it will remain in a test status and can still be edited. The document may be marked "READY FOR ISSUANCE" when the testing is complete and the document is ready for production.

Brief Summary Text (120):
As can be readily seen, the procedure for implementing dataset document changes is quite complex and involved. The prior art system requires entering data on multiple screens. The data that is entered is gathered after analysis, design, and testing have taken place. It is required that the same data be entered multiple times if multiple elements are being issued for the same project.

Brief Summary Text (125):
Enforcement of many standards is done when the issuance processor job runs. If any standards violations are encountered, the job is aborted. There are very few ways for the analysts to verify that they are meeting the standards when they are in test mode except to compare their element to the documentation in the DPM (Data Processing Manual) to ensure they are meeting standards.

Detailed Description Text (34):
Early capture of the metadata for use in cross-referencing, querying, etc. Today, the entry of the issuance metadata is `after the fact.` Since checkout, development, and testing are all done, there has been no audit trail of who is doing what before the element is moved into production. With RMS, the audit trail begins when the element is checked out for development, and continues throughout the development cycle and through promotion into production.

Detailed Description Text (62):
Element change information is then gathered directly form the developer or analyst in step S42. The analyst then checks out the modified element, and tests the modified element (in the workstation or host developer) in step S44. After the element has been satisfactorily tested and the analyst is satisfied that the element is ready to be distributed throughout the system to the appropriate components, a label is then assigned to the element to ensure that the appropriate version of the element is used to update the affected systems in step S46. The element is then checked in an area panvalet for initial archive and security purposes in step S48.

Detailed Description Text (63):
The analyst is then queried for additional changes in step S50, and if no additional changes are necessary, the element or elements that have been modified are uploaded to the host for storage/archival purposes in step S54. The element is then tested on the host in step S56. Assuming that the testing provides a satisfactory outcome in step S56, issuance information is provided to the librarian, analyst or even directly to BLDISS (or other comparable system) in step S58. The remaining steps of

the RMS check out/check in process have been discussed above.

Detailed Description Text (95):
A branch is a line of development. A branch is defined as a separate line of development consisting of one or more revisions that diverge from the trunk. Development branches start from a production revision on the trunk. A parallel branch of development can be initiated from a revision contained within a development branch. This allows development of alternate revisions of the same element for different projects. The most recent revision in a development branch is referred to as the tip.

Detailed Description Text (98):
To start the development of MCFUN.TXT, it must be checked out. Checking-out creates a new development branch. The first development branch from a trunk revision appends 1.0 to the trunk revision number creating a revision number of 1.0.1.0. If a second line of development were started, it would have 2.0 appended to the trunk revision creating a revision number of 1.0.2.0.

Detailed Description Text (99):
After checking-out the element, the resulting work file is edited in the appropriate development editor. FIG. 14 illustrates at position 110 the development branch number. Position 112 indicates the number of times the work file has been checked in.

Detailed Description Text (102):
In this example, the element has been completely tested and ready for issuance to production after five check-ins. When development revision 1.0.1.5 is issued to production, it will become the trunk revision 1.1. As new production revisions occur, the second position of the trunk revision number will be incremented by one (see position 120 of FIG. 18). FIG. 19 shows three branches of development on the MCFUN.TXT element after its first release to production.

Detailed Description Text (104):
A version label is a symbolic name assigned to the tip of a development branch. Version labels can only be used to retrieve the tips of development branches. This provides a convenient way to refer to a tip by a meaningful name. Revision numbers need not be remembered to check out elements. Instead, RMS references elements by their version labels. The types of versions labels are:

Detailed Description Text (108):
In FIG. 20, the work group name (PRS #) identifies the project and the revision number identifies the tip. The effective date of work group AX00498 is 9-8-96. The pilot date of revision 1.3.5.0 is 9-13-96. The Host version of the production revision is C00. The work group name of version 1.3.3.10 is CR 00900. Work group AXZ00498 was sourced from Revision 1.3.1.1. The work group name and effective date associated with the production revision are CM-15893, 8-1-96. Work groups GP-11594, AX-00499 and CR-00900 and tips 1.3.1.2, 1.3.2.1 and 1.3.3.10 are branched directly off the trunk. Revision numbers: 1.3.4.15 and 1.3.5.0 and Work groups AX-00498 and CM-15895 are branched from other branches. KEY=identifies the test revision by effective date, and PKEY=identifies the pilot key that is associated with tip 1.3.5.0.

Detailed Description Text (113):
A selection list shows available version labels from which may be chosen a source. A revision from the tree view may also be selected prior to selecting the check out option. Most of the time, a developer will initially check out an element from production. Subsequent check outs are usually done from work group name. Some areas regularly check out from other development branches, but special care must be taken when selecting this type of check out. The development version label will be the "KEY=. . . " field which contains the work group's effective date and work group number.

Detailed Description Text (123):
The Version Manager will be pointing to the working directory specified when the work group was created. Check out places a copy of the archive in this working

directory. Expanding the tree view of the archive will show that a new development branch has been created and is locked by your LAN logon ID. After checking out an element with a lock, the next step is to make changes to the work file in the working directory.

Detailed Description Text (156):
DEVELOPMENT BRANCH--a series of revisions that diverge from a trunk revision.

Detailed Description Text (204):
PROMOTION LEVEL--relates to different levels of testing. The bottom level, s where all source modification takes place, is always development and the top level is always production. The levels in between, if any, represent higher and higher levels of integration with existing production elements or other elements being developed concurrently.

Detailed Description Paragraph Table (1):
Version Label Description STATUS=PROD Identifies the latest production revision. KEY=key Identified the test revision by effective date. IKEY=key Identified the issuance key be effective date. IPKEY=key Identified the issuance key by pilot date. IRKEY=key Identifies the issuance key by Regional Support date. PKEY=key Identifies a test revision by pilot date. RKEY=key Identifies a test revision by Regional Support date. HOSTVER=X99 Identifies the three position host version, where X99 is an alpha followed by two numeric characters.